UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/806,261	03/22/2004	Kenneth Priddy	200314321-1	1933

22879 7590 08/06/2010

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER	
RIAD, AMINE	

ART UNIT	PAPER NUMBER
2113	

NOTIFICATION DATE	DELIVERY MODE
08/06/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
ipa.mail@hp.com
laura.m.clark@hp.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte KENNETH PRIDDY

Appeal 2009-005639
Application 10/806,261[1]
Technology Center 2100

Before JOHN A. JEFFERY, JEAN R. HOMERE, and JAMES R. HUGHES, *Administrative Patent Judges*.

HUGHES, *Administrative Patent Judge*.

DECISION ON APPEAL[2]

---

[1] Application filed March 22, 2004. The real party in interest is Hewlett-Packard Development Co., L.P. (App. Br. 3.)

[2] The two-month time period for filing an appeal or commencing a civil action, as recited in 37 C.F.R. § 1.304, or for filing a request for rehearing, as recited in 37 C.F.R. § 41.52, begins to run from the "MAIL DATE" (paper delivery mode) or the "NOTIFICATION DATE" (electronic delivery mode) shown on the PTOL-90A cover letter attached to this decision.

STATEMENT OF THE CASE

Appellant appeals from the Examiner's rejection of claims 1-12 under authority of 35 U.S.C. § 134(a). The Board of Patent Appeals and Interferences (BPAI) has jurisdiction under 35 U.S.C. § 6(b).

We reverse.

Appellant's Invention

Appellant invented a high-availability cluster and related method for reducing the downtime associated with the migration a RAM-intensive application from a failed computer to another computer in the high-availability cluster (a network of computers that collaborate to minimize interruptions due to component failures). Appellant's cluster and method execute instructions of a first instance of an application on a first computer to modify data stored in a volatile memory, create a snapshot of the data while the first instance of the application is running, store the snapshot in the volatile memory, and transfer the snapshot from the volatile memory to a storage media of the cluster. A second instance of the application program on a second computer accesses the snapshot stored in the storage media. (Spec. ¶¶ [02], [08].)[3]

Representative Claim

Independent claim 1 further illustrates the invention. It reads as follows:

---

[3] We refer to Appellant's Specification ("Spec."); Appeal Brief ("App. Br.") filed March 12, 2008; and Reply Brief ("Reply Br.") filed May 31, 2008. We also refer to the Examiner's Answer ("Ans.") mailed May 7, 2008.

1. A computer cluster comprising:

storage media;

a first computer having a first instance of an application program installed, said application program having instructions, said first computer including,

volatile memory;

processing means

for executing instructions of said first instance of said application program so as to modify data stored in said volatile memory,

for creating a snapshot of said data while said first instance of said application program is running, said snapshot being stored in said volatile memory, and

for, while said first instance of said application continues to modify said data so that it diverges from said snapshot, transferring said snapshot from said volatile memory to said storage media, and

a second computer having a second instance of said application program installed, said second computer including means for accessing said storage media so that said second instance of said application can access said snapshot as stored on said storage media.

Reference

The Examiner relies on the following reference as evidence of unpatentability:

Kelkar	US 7,058,846 B1	Jun. 6, 2006 (filed Oct. 17, 2002)

Rejection on Appeal

The Examiner rejects claims 1-12 under 35 U.S.C. § 102(b) as being anticipated by Kelkar.[4]

ISSUE

Based on our review of the administrative record, Appellant's contentions and the findings and conclusions of the Examiner, the pivotal issue before us is as follows:

Does the Examiner err in finding the Kelkar reference discloses creating a snapshot of application data while a first instance of an application program is running, storing the snapshot in volatile memory, and transferring the snapshot from the volatile memory to a storage media while the first instance of said application modifies the data so that it diverges from said snapshot?

FINDINGS OF FACT (FF)

Kelkar Reference

1. Kelkar describes a system and method enabling other nodes in a cluster to resume operations of a failed node. The system recognizes and immediately synchronizes resource configuration changes – changes to resource configuration data – among a set of nodes in a cluster. Then, even if a node that has made a resource configuration change fails, the resource configuration change is available for use by other nodes in the set, so that

[4] In the statement of the rejection (Ans. 3), the Examiner states that claims 1-14 are rejected. However, only claims 1-12 are pending. This appears to be a typographical error, uncontested by Appellant, which we find to be harmless.

each node in the set can resume operations of the failed node. (Col. 2, ll. 24-34; col. 3, ll. 33-44; col. 3, l. 55 to col. 4, l. 7; col. 6, ll. 5-42; Figs 1-3.)

2. Kelkar specifically describes a method for synchronizing configuration data:

> In action 3.1, a resource configuration is changed via resource configuration manager 360A. In action 3.2, resource configuration manager 360A updates resource attributes 372A of resource configuration data 370A. In action 3.3, resource configuration manager 360A notifies resource agent 314A for the affected resource. Resource configuration manager 360A notifies cluster manager 330A in action 3.4. Cluster manager 330A reads resource configuration data 370A and provides configuration data 305 to all nodes in the cluster in action 3.5 via cluster communication channel 215. Configuration data 305 may include all or a portion of resource configuration data 370A.
>
> In action 3.6, cluster manager 330B receives configuration data 305 indicating that resource attributes 372A are changed. In action 3.7, cluster manager 330B provides configuration data 305 to resource configuration manager 360B. In action 3.8, resource configuration manager 360B updates resource attributes 372B using configuration data 305 to reflect the changes made to resource attributes 372A. Resource configuration data 370A and 370B are now synchronized.

(Col. 6, ll. 7-27.)

3. Kelkar describes that the resource configuration changes may include "creating a snapshot of a storage area to back up data at given point in time." (Col. 3, ll. 43-44.) Kelkar also describes application programs installed on its nodes, and for example, nodes 110A and 110B may be configured as servers of the same application program. (Col. 5, ll. 8-9; col. 10, ll. 7-14.)

ANALYSIS

Appellant contends, *inter alia*, that the Kelkar reference does not disclose storing the snapshot in volatile memory (App. Br. 15; Reply Br. 10), or transferring the snapshot from volatile memory to storage media as the application program continues to modify data (App. Br. 16-17; Reply Br. 11-12). The Examiner finds that Kelkar discloses a first computer (node) with volatile memory and an application program executing instructions that modify data. The Examiner also finds that Kelkar discloses a storage media, a snapshot of the data stored in volatile memory, and transferring the snapshot to non-volatile memory. (Ans. 3-4, 7-8 (citing Kelkar, col. 3, ll. 33-36; col. 4, ll. 36-37, 44-46, 55-56; Figs. 1-3 & 7, elements 104A, 110A, 110B, 140, 140D, 370D, & 714).) Based on these contentions, we decide the question of whether the Examiner erred in finding the Kelkar reference discloses creating a snapshot of application data while an application program is running, storing the snapshot in volatile memory, and transferring the snapshot from the volatile memory to a storage media while the application modifies the data.

After reviewing the record on appeal, we agree with Appellant that the Kelkar reference does not disclose the disputed features of Appellant's claim 1. While we agree with the Examiner that Kelkar teaches the individual features of Appellant's claim, Kelkar fails to teach their interaction as recited.

We agree with the Examiner that Kelkar describes a storage media, a snapshot of data, and a first and a second node (computer) each having volatile memory with an application program executing instructions that modify data. (FF 1-3.) However, Kelkar fails to provide any description of

storing the snapshot in volatile memory of the first node or transferring the snapshot to the storage media, much less storing and transferring the snapshot as the application is running and modifying the data stored in memory.

The Examiner makes no specific factual finding as to the particular type of memory (in which Kelkar stores the snapshot), nor does the Examiner provide any citation (of Kelkar) supporting the position that the snapshot is stored in volatile memory. Rather, the Examiner summarily determines that Kelkar stores the configuration updates in volatile memory (as opposed to non-volatile memory (e.g., a disk drive)). (Ans. 4, 8.) Kelkar, however, does not explicitly describe the process of storing the configuration changes (snapshot). While it is apparent that Kelkar does store the configuration changes, it does not follow that the memory must be volatile, i.e., it is not inherent that the memory must be volatile. The fact that a certain result or characteristic may possibly occur or be present in the prior art is insufficient to establish the inherency of that result or characteristic. *See In re Rijckaert*, 9 F.3d 1531, 1534 (Fed. Cir. 1993).

> To establish inherency, the extrinsic evidence "must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill." "Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient."

In re Robertson, 169 F.3d 743, 745 (Fed. Cir. 1999) (citations omitted). "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic *necessarily* flows

from the teachings of the applied prior art." *Ex parte Levy*, 17 USPQ2d 1461, 1464 (BPAI 1990) (emphasis added). To anticipate, every element and limitation of the claimed invention must be found in a single prior art reference, arranged as in the claim. *Karsten Mfg. Corp. v. Cleveland Golf Co.*, 242 F.3d 1376, 1383 (Fed. Cir. 2001).

Kelkar describes making changes to a resource configuration (resource configuration data) in a node (computer) – which includes "creating a snapshot of a storage area to back up data at given point in time" (FF 3) – and storing the changes (in a storage or memory of the node). (FF 1-3.) For example, "[i]n action 3.2, resource configuration manager 360A updates resource attributes 372A of resource configuration data 370A." (FF 2; Kelkar, col. 6, ll. 8-10; Fig. 3.) Kelkar, however, does not describe storing the configuration changes (snapshot) in any particular type of storage or memory.

There is simply no explicit or inherent disclosure in Kelkar of storing configuration changes in a volatile memory, much less doing so while the recited application is running. Thus, we are constrained by the record before us, and find that Kelkar does not disclose creating a snapshot of application data while an application program is running and storing the snapshot in volatile memory.

Even if we were to agree with the Examiner's unsupported finding that Kelkar describes storing a snapshot in volatile memory, we cannot agree that Kelkar describes transferring the snapshot to a storage media (non-volatile memory). As with the volatile memory discussed *supra*, the Examiner makes no specific factual finding as to the storage device to which Kelkar transfers the snapshot. Rather, the Examiner summarily concludes

that Kelkar transfers the snapshot to a storage media (a non-volatile memory (e.g., a disk drive), as opposed to volatile memory (e.g., RAM)). (Ans. 4, 8.) The Examiner cites Kelkar's Storage Resource Definition (element 140D, Fig. 1; col. 4, ll. 44-46) as the non-volatile memory or storage media (Ans. 4, 8) – explaining that "figure 1 . . . shows 3 elements 104A, 102A, and 140D all in connection with storage resource 140" and "Examiner concludes that node 110A receives update of storage resource configuration at step 1.1 at element 104A, the update passes through step 1.2 from element 104a to element 102A, it then finally settles in item 140D by using step 1.3" and that "the update goes from volatile memory 104A to non volatile memory 140D" (Ans. 8). Kelkar, however, does not describe 140D as a non-volatile memory or a storage media.

While Kelkar does describe that 140D is associated with storage resource 140 – "updates storage resource definition 140D for storage resource 140" (Kelkar, col. 4, ll. 44-45) – Kelkar explains, in the next paragraph, that "[o]ne problem with system 100 described above is that storage resource definition 140D is stored on node 110A." Thus, we do not understand Kelkar to disclose transferring the configuration changes and storing them in storage resource (media) 140. Moreover, Kelkar explicitly describes transferring resource configuration data changes from the memory of a first node (computer) (element 370A of node 110A) to a memory of a second node (element 370B of node 110B). (FF 2.) Consequently, Kelkar does not disclose transferring the snapshot from the volatile memory to a storage media.

There is simply no explicit or inherent disclosure in Kelkar of transferring configuration data changes (a snapshot) from a volatile memory

to a storage media of a cluster (shared by multiple nodes or computers), much less doing so while the recited application is running and modifying data in the volatile memory. Thus, we are constrained by the record before us, and find that Kelkar does not disclose transferring the snapshot from the volatile memory to a storage media while the application modifies the data. Thus, Appellant has persuaded us to find error in the Examiner's anticipation rejection of claim 1.

Appellant's independent claim 7 includes the limitation "during a state that differs from the state represented in said snapshot, transferring said snapshot to storage media accessible by a second computer of said computer cluster." (App. Br. 27, Claim App'x, claim 7.) This limitation is similar in scope to the transferring limitation of claim 1 (*supra*). It follows, for the reasons discussed with respect to claim 1 *supra*, that Kelkar does not disclose transferring a snapshot to storage media accessible by a second computer during a memory state that differs from that of the snapshot (i.e., when the data of the memory has changed or is changing). Appellant's dependent claims 2-6 and 8-12 depend on the respective base independent claims (claims 1 and 7).

Therefore, based on the record before us, we find that the Examiner erred in finding that the Kelkar reference discloses each of the disputed features of Appellant's independent claim 1 and dependent claims 2-6. We reach this same conclusion regarding Appellant's independent claim 7 and dependent claims 8-12, which recites commensurate limitations. Accordingly, we reverse the Examiner's anticipation rejection of claims 1-12.

CONCLUSION OF LAW

On the record before us, we find Appellant has shown that the Examiner erred in rejecting claims 1-12 under 35 U.S.C. § 102(b).

DECISION

We reverse the Examiner's rejections of claims 1-12 under 35 U.S.C. § 102(b).

REVERSED

rwk

Hewlett Packard Company
P O Box 272400, 3404 E. Harmony Road
Intellectual Property Administration
Fort Collins, CO 80527-2400